3U CompactPCI ビデオキャプチャ

aPCI-8701

ユーザーズマニュアル

株式会社 アドテック システム サイエンス

保証規定

保証の範囲

この保証規定は、弊社―株式会社アドテックシステムサイエンスが製造・出荷し、お客様にご購入いただいたハードウェア製品に適用されます。弊社によって出荷されたソフトウェア製品については、弊社所定のソフトウェア使用許諾契約書の規定が適用されます。

弊社以外で製造されたハードウェア又はソフトウェア製品については、製造元／供給元が出荷した製品そのままで提供いたしますが、このような製品には、その製造元／供給元が独自の保証を規定することがあります。

保証条件

弊社は、以下の条項に基づき製品を保証いたします。不慮の製品トラブルを未然に防ぐためにも、あらかじめ各条項をご理解のうえ製品をご使用ください。

この保証規定は弊社の製品保証の根幹をなすものであり、製品によっては、その取扱説明書や保証書などで更に内容が細分化され個別に規定されることがあります。したがって、ここに規定する各条項の拡大解釈による取扱いや特定目的への使用に際しては十分にご注意ください。

製品の保証期間は、製品に添付される「保証書」に記載された期間となり、弊社は、保証期間中に発見された不具合な製品について保証の責任をもちます。

保証期間中の不具合な製品について、弊社は不具合部品を無償で修理又は交換します。ただし、次に記載する事項が原因で不具合が生じた製品は保証の適用外となります。

— 事故、製品の誤用や乱用
— 弊社以外が製造又は販売した部品の使用
— 製品の改造
— 弊社が指定した会社以外での調整や保守、修理など

弊社から出荷された後に災害又は第三者の行為や不注意によってもたらされた不具合及び損害や損失については、いかなる状況に起因するものであっても弊社はその責任を負いません。

原子力関連、医療関連、鉄道等運輸関連、ビル管理、その他の人命に関わるあらゆる事物の施設・設備・器機など全般にわたり、製品を部品や機材として使用することはできません。もし、これらへ使用した場合は保証の適用外となり、いかなる不具合及び損害や損失についても弊社は責任を負いません。

特別な取り扱いについての注意

aPCI-8701 回路基板には、静電放電(ESD)の影響をうけやすい CMOS 回路が含まれています。aPCI-8701 の取り扱い、輸送、取り付けに際しては、静電放電による回路基板の損傷を防止するために特別の注意が必要です。とくに、下記を順守してください。

— 回路基板をエンクロージャに取り付ける準備ができるまで、静電気防止パッケージから回路基板を取り出さないでください。

— アース（グランド）を取った静電放電防止ステーション以外では、回路基板を取り扱わないでください。

— 回路基板の取り付け、取り外しの前に、CopmpactPCI バスから電源を切り離してしてください。

目次

1. はじめに

aPCI-8701 CompactPCI ビデオキャプチャを使用して、多様なアナログビデオソースからコンピュータのメモリ (RAM) にモノクロ およびカラーイメージを取り込むことができます。CompactPCI バス用の設計になっています。IBM PC および互換機用に設計されたソフトウェアが搭載されています。

aPCI-8701 は、信号ソースの種類を問わず、入力ビデオ信号をデジタルに変換して、安定した出力を提供します。入力の 3:1 マルチプレクサにより、2 本のコンポジット、1 本の Y/C (S-Video)アナログビデオ入力の間でソース切り替えができます。つまり最大 3 種類のビデオソースをビデオキャプチャ（以下フレームグラバ）に接続することができます。aPCI-8701 は、入力ビデオフォーマット (NTSC、PAL または SECAM) を自動的に検出して、コンポジット信号の輝度 (Y) とクロミナンス (C) 成分を分離します。それぞれの信号成分は、2 つ別々の 8 ビット A/D コンバータでデジタル化されます。入力信号にローパスフィルタリングとダブルオーバサンプリングをかけることで、エイリアシングアーチファクトの発生が完全に排除され、精密にデジタル化されます。そのあと、デジタル信号が、必要に応じて希望の寸法にスケーリング/クロッピングされます。スケーリングされたイメージはバーストモードを使ってホストの RAM に転送されます。オンボードの FIFO で、必要なバッファリングが確保され、イメージ損失の確率が最小に抑制されます。

aPCI-8701 と 外部ハードウェア間のインタフェースには 8 ビット汎用 I/O ポートが用意されています。これを、たとえばトリガ付きイメージ取り込みに利用することもできます。

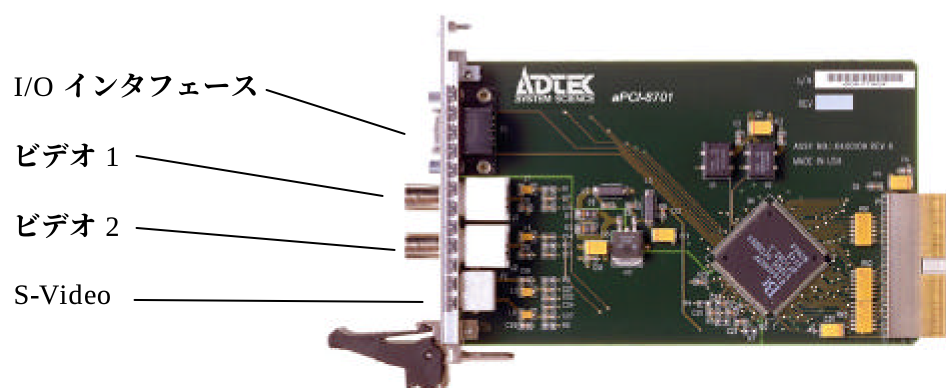
2. システム要件

aPCI-8701 は、3U、32 ビット、33MHz の CompactPCI スロットを 1 スロット必要とします。486CPU、8MB の RAM が最低要件ですが、16MB の Pentium 機を使用することを推奨します。

3.　仕様

パラメータ	仕様
ビデオソース	NTSC、PAL、SECAM、RS-170、CCIR
ビデオ入力	アナログコンポジットビデオ (BNC) × 2 アナログ Y/C (S-Video) (DIN) × 1
出力フォーマット	RGB (24 ビット)、Y8 (8 ビット)
出力解像度 (最大)、ピクセル値	754× 480 (NTSC、RS-170) 922× 576 (PAL、SECAM、CCIR)
A/D 分離 輝度チャンネル クロミナンスチャンネル	 8 ビット 8 ビット
キャプチャレート	リアルタイム： 30fps (NTSC、RS-170) 25fps (PAL、SECAM、CCIR)
汎用 I/O ポート	入力系統 4、出力系統 4、TTL/CMOS、DB15 雌コネクタ
バス要件	CompactPCI 3U スロット (33MHz、32 ビット) × 1
消費電力	200mA (最大) @ + 5V 100mA (最大) @ + 12V
動作温度	0～70℃

3.1　コネクタ配置図



3.2　汎用 I/O ポートのコネクタのピン配置

ピン番号	信号	ピン番号	信号
1	OUT1	11	OUT3
3	OUT0	13	INP0
5	INP3	15	INP1
7	OUT2	4, 6, 8, 12	GND
10	INP2	2, 9, 14	n/c

4. ソフトウェアレファレンス

aPCI-8701 には、SX11 ソフトウェア開発キット (SDK) が同梱されています。SDK には、Windows 95 または Windows NT 上で aPCI-8701 を使ってイメージングアプリケーションを作成するのに必要な各種コンポーネントのほか、ソフトウェアドキュメンテーション、サンプルアプリケーション (ソースコード付き) が入っています。コンピュータに SX11 SDK をインストールするには、配布キットのディスク 1 から `setup.exe` を実行します。コンピュータに次のファイルがインストールされます。

`\HELP\sx11.hlp` - オンラインヘルプ (データの型と DLL 関数についての詳細な説明を含みます)

`\INCLUDE\sx11.h` - C include ファイル (型と定数)

`\INCLUDE\sx11f.h` - C include ファイル (関数のプロトタイプ)

`\INCLUDE\sx11.ico` - SX11 アイコン
`\INCLUDE\sx11_vbi.bas` - Visual Basic 用の "include"ファイル

`\LIBRARY\sx11.dll` - 32 ビットダイナミックリンクライブラリ (Windows 95 と NT 用)
`\LIBRARY\sx11.lib` - インポートライブラリ (C アプリケーション用)

```
\SAMPLES\SAMPLE1\...
\SAMPLES\SAMPLE2\...
\SAMPLES\SAMPLE3\...
```

これらのディレクトリには、イメージ取り込みのテクニックを例証する 3 種類のサンプルアプリケーションのソースファイルと実行ファイルが入っています。アプリケーションは C 言語で書かれています。サンプルアプリケーションを変更するには、project ファイルを作成し、(アイコンやインポートライブラリファイルを含めて) すべての必要なコンポーネントを指定する必要があります。サンプルは、Microsoft Visual C++ v.4.0、Watcom C/C++ v.11.0 で正常コンパイルのテスト済みです。

`\SAMPLES\VBSAMPLE\SETUP...`

Visual Basic サンプルインストール用セットアップファイル。本体のインストールが終了したあと、このディレクトリから、`setup.exe` を実行します (または、SK11_SDK フォルダの Setup アイコンをダブルクリックします)。VB サンプルアプリケーションを実行するのに必要なコンポーネントがコンピュータにインストールされます。Microsoft Visual Basic v.4.0 で書かれたサンプルアプリケーションのソースファイルが`\SAMPLES\VBSAMPLE\SOURCE` にインストールされます。サンプルアプリケーションを変更するには、project ファイルを作成し、(アイコンや VB "include"ファイルを含めて) すべての必要なコンポーネントを指定する必要があります。

\SAMPLES\MILSAMPL\...

Matrox Imaging Library (MIL) へのインタフェースの例としてのサンプルアプリケーション

ダイナミックリンクライブラリ sx11.dll のインストール先ディレクトリ

\WINDOWS\SYSTEM - Windows 95 用
\WINNT\SYSTEM32 - Windows NT 用

sx11.dll のコピーが \LIBRARY に格納されています。

上記のファイルに加えて、SDK から次のドライバコンポーネントがインストールされます。

\WINDOWS\SYSTEM\VMM32\windrvr.vxd - Windows 95 用
\WINNT\SYSTEM32\DRIVERS\windrvr.sys - Windows NT 用

SX11.hlp の内容は付録 A を参照してください 。

5. テクニカルサポート

テクニカルサポートについては、弊社テクニカルセンターにお問い合わせください。

TEL：　045-333-0335
FAX：　045-331-7770
E-mail：　support@adtek.co.jp
ホームページ：　http://www.adtek.co.jp/

付録 A： SX11.dll データの型と関数

データの型

フレームグラバハンドル HFG

フレームグラバ用のハンドルとして使用される 32 **ビット値**

イメージバッファハンドル HBUF

イメージバッファ用のハンドルとして使用される 32 **ビット値**

エラーコード ECODE

エラーコードとして使用される 32 **ビット値**

システムデータ構造体 PCI

```
typedef struct {
     DWORD boards;
     DWORD PCIslot[SYS_GRABBERS];
} PCI;
```

PCI 構造体には、システムの識別したフレームグラバボードについての情報が格納されます。

メンバ	記述
boards	システムの識別した、サポートされているフレームグラバボードの数
PCIslot	スロット番号の配列

PCI 構造体は、X11_InitSystem 関数で初期化されます。システム定数 SYS_GRABBERS は、サポートされているフレームグラバの最大数を決定します。値は *SX11.h* で定義されます。メンバ PCIslot **は、特定ボードの** PCI **スロット番号を表します。**PCI **スロット番号は、**PCI BIOS **で生成され、通常のスロット番号と同じでない場合もあります。**

フレームデータ構造体 FRAME

```
typedef struct {
     LPBITMAPINFO lpbmi;
     void * lpvbits;
} FRAME;
```

FRAME 構造体は、システムにより割り当てられたイメージバッファの個々のフレームについての情報が格納されます。

メンバ	記述
lpbmi	フレームに関連付けされたデバイス独立ビットマップ (DIB) の寸法と色情報を定義する BITMAPINFO 構造体へのポインタ
lpvbits	フレームのイメージデータへのポインタ

イメージバッファが割り当てられると、すべてのフレームが Windows DIB に関連付けされ、対応するデータ構造体が作成されます。これは、Windows API を使って、取り込まれたイメージの表示を単純化します。ただし、lpvbits をフレームデータへの汎用ポインタとして使うことも可能です。ビットマップ、関連データ構造体、表示関数の説明については Windows API のヘルプを参照してください。

注記：
バッファが「フラットな」バッファとして作成される場合でも、BITMAPINFO 構造体が作成されます (MODE 構造体の store メンバの説明を参照してください)。この場合、Windows API 関数が、イメージを上下さかさまに表示します。

イメージバッファデータ構造体 BUFFER

```
typedef struct {
    HBUF hbuf;
    DWORD dwFrames;
    FRAME frame[SYS_FRAMES];
} BUFFER;
```

BUFFER 構造体は、システムの割り当てたイメージバッファについての情報を格納します。イメージバッファは、1 **つまたは複数のイメージフレームで構成されます。取り込み関数は、常に、連続的に取得** (grab) **されたビデオフレームに対応するデータで、**1 **つのイメージバッファのすべてのフレームを埋めます。**

メンバ	記述
hbuf	**イメージバッファへのハンドル**
dwFrames	**イメージバッファ内のフレームの数**
frame[SYS_FRAMES]	**FRAME 構造体の配列**

BUFFER 構造体は、X11_AllocateBuffer 関数で初期化されます。

オペレーションモードデータ構造体 MODE

```
typedef struct {
    DWORD scale;
    DWORD color;
    DWORD store;
    DWORD input;
    MODE_ADVANCED advanced;
} MODE;
```

MODE 構造体は、フレームグラバのオペレーションについての情報を格納します。ひんぱんに使用される見込みのないセッティングは、advanced メンバ内部に隠蔽されます。

メンバ	記述
scale	イメージのスケールを定義します。次のいずれかになります。
color	イメージの出力カラーフォーマットを定義します。次のいずれかになります。
store	イメージをどのようにメモリに格納するかを定義します。 次のいずれかになります。
input	入力マルチプレクサを制御します。次のいずれかになります。
advanced	MODE_ADVANCED 構造体。アドバンストモードの設定を定義します。

scale:

値	記述
SCALE_ADVANCED	イメージスケールは、MODE_ADVANCED 構造体の設定で定義されます。
SCALE8	フルサイズイメージ
SCALE6	3/4 サイズのイメージ
SCALE4	1/2 サイズのイメージ
SCALE2	1/4 サイズのイメージ

color:

値	記述
COLOR_MONO	モノクロイメージ、1 バイト/ピクセル
COLOR_RGB	カラーイメージ、3 バイト/ピクセル

store:

値	記述
STORE_DIB	イメージは Windows DIB (ラインオーダの上下反転) として格納されます。
STORE_FLAT	イメージは通常のラインオーダとして格納されます。

input:

値	記述
MUX_0	S-Video 入力
MUX_1	ビデオ 1 入力
MUX_2	ビデオ 2 入力

アドバンストオペレーションモードデータ型 MODE_ADVANCED

```
typedef struct {
     DWORD interlace;
     DWORD xTotal;
     DWORD xActive;
     DWORD xDelay;
     float yFactor;
     DWORD yActive;
     DWORD yDelay
     DWORD FORMAT;
     DWORD BRIGHT;
     DWORD CONTRAST;
     DWORD SAT_U;
     DWORD SAT_V;
     DWORD HUE;
     DWORD LNOTCH;
     DWORD LDEC;
     DWORD DEC_RAT;
     DWORD PEAK;
     DWORD CAGC;
     DWORD CKILL;
     DWORD HFILT;
     DWORD RANGE;
     DWORD CORE;
     DWORD YCOMB;
     DWORD CCOMB;
     DWORD ADELAY;
     DWORD BDELAY;
     DWORD SLEEP;
     DWORD CRUSH;
     DWORD VFILT;
     DWORD COLOR_BARS;
     DWORD GAMMA;
     DWORD PKTP;
} MODE_ADVANCED;
```

MODE_ADVANCED 構造体は、フレームグラバのアドバンストオペレーションモードについての情報を格納します。

<table>
<tr><th>メンバ</th><th>記述</th></tr>
<tr><td>interlace</td><td>入力イメージフォーマットを定義します。次のいずれかになります。
<table>
<tr><th>値</th><th>記述</th></tr>
<tr><td>IMG_INTERLACED</td><td>インターレース入力イメージ</td></tr>
<tr><td>IMG_NONINTERLACED</td><td>ノンインターレース入力イメージ</td></tr>
</table></td></tr>
<tr><td>xTotal</td><td>(水平ブランキングを含む)出力水平ピクセルの合計数。有効値は 100～910 (NTSC) または 1135 (PAL)。フレームグラバにより内部的に生成されるピクセルの数です。</td></tr>
<tr><td>xActive</td><td>アクティブ出力水平ピクセルの合計数。有効値は 80～900 (NTSC) または 1000 (PAL)。出力イメージのピクセル数です。</td></tr>
<tr><td>xDelay</td><td>水平同期に対する、水平方向のアクティブ領域開始オフセット、ピクセル値。常に、次の条件が充足されなければなりません： xDelay + xActive <= xTotal</td></tr>
<tr><td>yFactor</td><td>垂直スケーリング係数。ビデオソースが生成する実際のライン数 (NTSC は 525、PAL は 625) を yFactor で除した値が出力イメージのライン数になります。有効値は 1.0～8.0。</td></tr>
<tr><td>yActive</td><td>(垂直スケーリング係数が適用される前の)アクティブ出力ラインの合計数。有効値は 60～525 (NTSC) または 625 (PAL)。たとえば、係数 2 でスケールダウンした NTSC イメージ全体を取得 (grab) するには、yActive を 525 に、yFactor を 2.0 に設定します。係数 2 でスケールダウンした NTSC イメージの上半分を取得 (grab) するには、yActive を 262 に、yFactor を 2.0 に設定します。</td></tr>
<tr><td>yDelay</td><td>垂直同期との相対で、垂直方向のアクティブ領域開始オフセット、(垂直スケーリングが適用される前の)ライン数。常に、次の条件が充足されなければなりません： yDelay + yActive <= (合計行数、525 または 625)</td></tr>
<tr><td>FORMAT</td><td>入力信号フォーマット。次のいずれかになります。
<table>
<tr><th>値</th><th>記述</th></tr>
<tr><td>FORMAT_NTSC</td><td>NTSC 入力信号</td></tr>
<tr><td>FORMAT_NTSCJ</td><td>NTSC (日本) 入力信号</td></tr>
<tr><td>FORMAT_PAL</td><td>PAL 入力信号</td></tr>
<tr><td>FORMAT_PALM</td><td>PAL (M) 入力信号</td></tr>
<tr><td>FORMAT_PALN</td><td>PAL (N) 入力信号</td></tr>
<tr><td>FORMAT_PALNC</td><td>PAL (N コンビネーション) 入力信号</td></tr>
<tr><td>FORMAT_SECAM</td><td>SECAM 入力信号</td></tr>
</table></td></tr>
<tr><td>BRIGHT</td><td>出力信号の明度 (輝度) を制御します。0～0xFF の値をとります。値は -128 (0×80) ～ +127 (0×7F) の符号付きオフセットとみなされます。輝度変化の解像度は 1LSB (フルレンジの 0.4%)。</td></tr>
<tr><td>CONTRAST</td><td>この 9 ビット値に輝度値を乗じて、コントラスト (ゲイン) 調整を得ます。0～0×1FF (237%) の値をとります。0×0D8 が 100%に対応します。</td></tr>
</table>

SAT_U ビデオ信号の U 成分にゲイン調整を追加するのに使用される 9 ビット値。同じインクリメント値ずつ U および V カラー成分を調整することにより、クロミナンスが調整されます。0〜0×1FF (201%) の値をとります。0×0FE が 100%に対応します。

SAT_V ビデオ信号の V 成分にゲイン調整を追加するために使用される 9 ビット値。同じインクリメント値ずつ U および V カラー成分を調整することにより、クロミナンスが調整されます。0〜0×1FF (284%)の値をとります。0×0B4 が 100%に対応します。

HUE 復調副搬送波フェーズの調整により色相を制御します。0〜0×FF の値をとります。値は符号付きオフセットとみなされます。0×80 は -90 度、0×7F は +89 度に対応します。

LNOTCH コンポジット信号を使用するケースで、出力イメージの「格子模様」パターンを除去して、出力信号の副搬送波を減衰させる内部輝度ノッチフィルタを制御します。次のいずれかになります。

値	記述
LNOTCH_OFF	フィルタ無効化
LNOTCH_ON	フィルタ有効化

LDEC 輝度信号の高周波数成分を削減するのに使用される輝度デシメーション (decimation) フィルタを制御します。より低い解像度にスケールダウンするときに有効です。詳しくは、HFILT を参照してください。次のいずれかになります。

値	記述
LDEC_OFF	フィルタ無効化
LDEC_ON	フィルタ有効化

DEC_RAT 60 (NTSC) または 50 (PAL/SECAM) のドロップアウトフィールドまたはフレーム数に対応する 6 ビット値。値 0 はデシメーションを無効にします。

PEAK 標準またはピーク輝度ローパスフィルタを HFILT 経由で実装するかどうかを指定します。次のいずれかになります。

値	記述
PEAK_OFF	標準ローパスフィルタ
PEAK_ON	ピークローパスフィルタ

CAGC クロミナンス AGC 機能を制御します。有効にすると、入力クロミナンス信号に 0.5〜2.0 の範囲の値を値を乗じることにより標準外のクロミナンスレベルを補正します。次のいずれかになります。

値	記述
CAGC_OFF	Chroma AGC 無効化
CAGC_ON	Chroma AGC 有効化

CKILL　ローカラーの検出、除去回路を制御します。次のいずれかになります。

値	記述
CKILL_OFF	ローカラーの検出、除去を有効化
CKILL_ON	ローカラーの検出、除去を無効化

HFILT　LDEC が LDEC_ON に設定されている場合に、水平ローパスフィルタリングの程度を制御します。次のいずれかになります。

値	記述
HFILT_AUTO	スケールの設定に応じて、自動的にフィルタが選択されます。水平スケーリングがフルから 1/2 までのとき、フィルタリングなしになります。スケーリングが 1/2 から 1/4 のとき、CIF フィルタが使用されます。1/4 から 1/8 のときは、QCIF フィルタが使用されます。1/8 より小さいスケーリングでは、ICON フィルタが使用されます。
HFILT_CIF	CIF フィルタ
HFILT_QCIF	QCIF フィルタ
HFILT_ICON	ICON フィルタ

RANGE　輝度出力の範囲を決定します。次のいずれかになります。

値	記述
RANGE_NORM	標準オペレーション (輝度範囲 16～253)
RANGE_FULL	フルレンジオペレーション (輝度範囲 0～255)

CORE　コアリング値を制御します。コアリングを有効にすると、特定値未満の輝度レベルは切り捨てられて 0 になります。次のいずれかになります。

値	記述
CORE_OFF	コアリング無効化
CORE_8	コアリング閾値を 8 に設定
CORE_16	コアリング閾値を 16 に設定
CORE_24	コアリング閾値を 24 に設定

YCOMB　輝度コムフィルタリングを制御します。次のいずれかになります。

値	記述
YCOMB_OFF	垂直ローパスフィルタリングと垂直補間
YCOMB_ON	垂直ローパスフィルタリングのみ。フィルタタップの数は VFILT の設定で指定されます。

CCOMB　クロミナンスコムフィルタリングを制御します。次のいずれかになります。

値	記述
CCOMB_OFF	クロミナンスフィルタリングを無効化
CCOMB_ON	クロミナンスフィルタリングを有効化

ADELAY バックポーチ (複合画像信号) サンプリング遅延。デフォルト値は 0× 68 (NTSC) と 0× 7F (PAL/SECAM)

BDELAY 副搬送波サンプリング遅延。デフォルト値は 0× 5D (NTSC) と 0× 73 (PAL/SECAM)

SLEEP 輝度、クロミナンス A/D のスリープモードを制御します。次のいずれかになります。

値	記述
SLEEP_OFF	A/D の両方を有効化
Y_SLEEP	輝度の A/D をスリープモードにする。
C_SLEEP	クロミナンスの A/D をスリープモードにする。A/D の両方を無効化するには、**Y_SLEEP** と **C_SLEEP** の論理和を使うことができます。

CRUSH AGC モードを制御します。次のいずれかになります。

値	記述
CRUSH_OFF	非適応 AGC
CRUSH_ON	適応 AGC。A/D のオーバフローに対応して、入力電圧範囲が上昇します。

VFILT 垂直スケーリングフィルタのタップ数を制御します。次のいずれかになります。

値	記述
YCOMB が YCOMB_ON に設定されているとき	
VFILT_0	2 タップフィルタ
VFILT_1	3 タップフィルタ。385 水平アクティブピクセル未満のスケーリングの場合のみ利用可能
VFILT_2	4 タップフィルタ。193 水平アクティブピクセル未満のスケーリングの場合のみ利用可能
VFILT_3	5 タップフィルタ。193 水平アクティブピクセル未満のスケーリングの場合のみ利用可能
YCOMB が YCOMB_OFF に設定されているとき	
VFILT_0	2 タップ補間のみ
VFILT_1	2 タップフィルタと 2 タップ補間。385 水平アクティブピクセル未満のスケーリングの場合のみ利用可能
VFILT_2	3 タップフィルタと 2 タップ補間。193 水平アクティブピクセル未満のスケーリングの場合のみ利用可能
VFILT_3	4 タップフィルタと 2 タップ補間。193 水平アクティブピクセル未満のスケーリングの場合のみ利用可能

COLOR_BARS テストカラーバーのパターンを制御します。次のいずれかになります。

値	記述
COLORBARS_OFF	カラーバーをオフにします。
COLORBARS_ON	カラーバーをオンにします。

GAMMA ガンマ補正除去を制御します。次のいずれかになります。

値	記述
GAMMA_REMOVE_ON	ガンマ補正除去をオンにします。
GAMMA_REMOVE_OFF	ガンマ補正除去をオフにします。

PKTP FIFO トリガポイント。次のいずれかになります。

値	記述
PKTP4	4DWORD
PKTP8	8DWORD
PKTP16	16DWORD
PKTP32	32DWORD

割り込みデータ型 INT_DATA

```
typedef struct {
     HFG hfg;
     DWORD mask;
     DWORD status;
     FPTR func;
     int priority;
     DWORD total;
     DWORD lost;
} INT_DATA;
```

INT_DATA 構造体は、割り込みのサポートに必要な情報を格納します。

メンバ	記述
hfg	フレームグラバハンドル。フレームグラバを選択します。
mask	割り込みマスク。**mask** を 0 に設定すると割り込みが無効になります。次の値に mask を設定すると (論理和を使用できます)、指定の割り込みが有効になります。

値	記述
STATUS_READY	フレーム取り込みが完了
STATUS_VIDEO	入力でビデオステータス変更 (例：present から absent へ)
STATUS_HLOCK	入力で水平ロック状態変更
STATUS_OFLOW	オーバフロー検出
STATUS_HSYNC	新しいラインの開始
STATUS_VSYNC	新しいフィールドの開始
STATUS_FMT	ビデオフォーマット変更の検出 (例：NTSC から PAL へ)
STATUS_ERROR	転送エラー発生。これはビットの組み合わせです。

メンバ	記述
status	フレームグラバのステータス。個別ビットの意味は、割り込みマスクのそれに対応します。割り込み発生時に **status** の値が設定されます。
func	ユーザ割り込み処理関数へのポインタ。関数の戻り値は DWORD で、引数はとりません。
priority	スレッドの優先順位レベルを処理する割り込みを指定する整数。次の値をとります。 **THREAD_PRIORITY_LOWEST** **THREAD_PRIORITY_BELOW_NORMAL** **THREAD_PRIORITY_NORMAL** **THREAD_PRIORITY_ABOVE_NORMAL** **THREAD_PRIORITY_HIGHEST** **THREAD_PRIORITY_TIME_CRITICAL**

total	**X11_InterruptOn呼び出し後に発生した割り込みの合計数**
lost	**処理されなかった割り込みの数**

注記

割り込みマスクのビットはステータスワードのビットに対応しています。したがって、個々のステータス条件の選択に同じ定数が使えます。

関数

X11_InitSystem

__declspec(dllimport) ECODE__stdcall X11_InitSystem (*pPCIdata***)**

PCI * *pPCIdata*;   /* **PCI 構造体のアドレス** */

X11_InitSystem は、システム初期化関数です。システム上でサポートされているボードを識別し、内部データ構造を初期化します。

パラメータ	記述
pPCIdata	PCI 型変数へのポインタ。

戻り値

初期化に成功すると 0**、それ以外はエラーコードを戻します。また、**PCI **型構造体を変更します。boards には、識別された、サポートされているボードの数が格納されます。PCIslot 配列には識別されたボードのスロット数が格納されています。各ボードのフレームグラバハンドルは、X11_GetHFG 関数を使って取得することができます。**

例

```
PCI pci;
ECODE ecode;
int i;
if (!(ecode = X11_InitSystem (&pci))) {
    printf (“Boards %d\n”, pci.boards);
    for (i = 0; i < pci.boards; i++) {
        printf (“Slot %04X\n”, pci.PCIslot[i]);
    }
}
else {
    return ecode;
}
```

X11_CloseSystem

__declspec(dllimport) void__stdcall X11_InitSystem (void)

X11_CloseSystem 関数は、現在のシステムリソース割り当てを解放します。アプリケーション終了前にかならず呼び出さなければなりません。

例

```
PCI pci;
ECODE ecode;
if (!(ecode = X11_InitSystem (&pci))) {
    /* application code here */
    X11_CloseSystem;
    return 0;
}
else {
    return ecode;
}
```

X11_AllocateBuffer

__declspec(dllimport) ECODE__stdcall X11_AllocateBuffer (*pMode, pBuffer, frames***)**

MODE * *pMode*; /* **MODE 構造体のアドレス** */
BUFFER * *pBuffer*; **/* BUFFER 構造体のアドレス */**
DWORD *frames*; /* **バッファ内のフレーム数** */

X11_AllocateBuffer 関数は、イメージバッファを割り当てます。

パラメータ	記述
pMode	**MODE 型変数へのポインタ。バッファのサイズとプロパティを定義するには、X11_AllocateBuffer 関数の呼び出し前に MODEがセットアップされていなければなりません。スケーリング、カラーフォーマット、ストレージタイプに影響する MODE の設定を変更する場合は、イメージバッファの再割り当てが必要になります** (**MODE、MODE_ADVANCED、X11_FreeBuffer を参照してください**)。
pBuffer	**BUFFER 型変数へのポインタ。呼び出しに成功すると、X11_AllocateBuffer 関数により、BUFFER 構造体のメンバが設定されます。**
frames	**イメージバッファ内のフレームの数。かならず、**1 **〜 SYS_FRAMES の間の整数値です。**

戻り値

成功すると 0**、それ以外はエラーコードを戻します。また、BUFFER 型構造体を変更します。hbuf には、割り当てられているイメージバッファハンドルが格納されます。dwFrames には、イメージバッファ内のフレーム数、frame 配列には、個々のフレームのデータおよび関連付けされたビットマップ情報へのポインタが格納されます。**

例

```
PCI pci;
ECODE ecode;
MODE mode = {DEF_MODE};
BUFFER buffer;

if (!(ecode = X11_InitSystem (&pci))) {
    if (!(ecode = X11_AllocateBuffer (&mode, &buffer, 1))) {
        /* application code here */
    }
    else {
        return ecode;
    }
}
else {
    return ecode;
}
```

X11_AllocateBufferExternal

__declspec(dllimport) ECODE__stdcall X11_AllocateBufferExternal
(*pMode, pBuffer, frames, pXbuf***)**

```
MODE * pMode;        /* MODE 構造体のアドレス */
BUFFER * pBuffer:    /* BUFFER 構造体のアドレス */
DWORD frames;        /* バッファ内のフレーム数 */
void * pXbuf;        /* 外部バッファのアドレス */
```

X11_AllocateBufferExternal 関数は、イメージバッファを外部関数が提供するバッファにマッピングします。

パラメータ	記述
pMode	MODE 型変数へのポインタ。バッファのサイズとプロパティを定義するには、X11_AllocateBufferExternal 関数の呼び出し前に MODEがセットアップされていなければなりません。バッファのディメンションとバイト/ピクセル値が正確に外部バッファのそれに一致していなければなりません。
pBuffer	BUFFER 型変数へのポインタ。呼び出しに成功すると、X11_AllocateBufferExternal 関数により、BUFFER 構造体のメンバが設定されます。
frames	イメージバッファ内のフレームの数。かならず、1 ～ SYS_FRAMES の間の整数値です。ほとんどの場合、1 です。

戻り値

成功すると 0、それ以外はエラーコードを戻します。

注記

1. この関数は、*SX11.dll* と他のイメージングアプリケーション高速インタフェースに対応するように設計されています。*SX11.dll* からアクセス可能なイメージバッファを、別のアプリケーションまたはライブラリの割り当てるバッファにマッピングします。たとえば、データをコピーする必要なしに、イメージ処理アプリケーションのイメージバッファに直接フレームを取り込むことができます。

2. X11_FreeBuffer と X11_CloseSystem 関数は、X11_AllocateBufferExternal 関数の割り当てたバッファに属するメモリを解放しません。メモリを割り当てたアプリケーションから解放する必要があります。

X11_FreeBuffer

__declspec(dllexport) void__stdcall X11_FreeBuffer (*hbuf*)

HBUF *hbuf;* /* **イメージバッファハンドル** */

X11_FreeBuffer 関数は、イメージバッファに関連付けされたリソースを解放します。

パラメータ	記述
hbuf	解放するイメージバッファのハンドル

注記

(イメージフォーマットを変更するときなど） イメージバッファを再割り当てする場合は、X11_FreeBuffer を呼び出す必要があります。アプリケーションを終了するときは、X11_CloseSystem 関数がバッファリソースを解放するため、X11_FreeBuffer を呼び出す必要はありません。

例

```
PCI pci;
ECODE ecode;
MODE mode = {DEF_MODE};
BUFFER buffer;

if (!(ecode = X11_InitSystem (&pci))) {
    if (!(ecode = X11_AllocateBuffer (&mode, &buffer, 1))) {
        /* application code here */
        /* now we change the scaling */
        mode.scale = SCALE2;
        X11_FreeBuffer (buffer.hbuf);
        /* re-allocate the buffer */
        if (!(ecode = X11_AllocateBuffer (&mode, &buffer, 1))) {
         /* application code here*/
        }
        else {
         return ecode;
        }
    }
    else {
        return ecode;
    }
}
else {
    return ecode;
}
```

__declspec(dllimport) ECODE__stdcall X11_Acquire (*hfg, hbuf, timeout, pStatus***)**

HFG *hfg*;	/* **フレームグラバハンドル** */
HBUF *hbuf*;	/* **イメージバッファハンドル** */
float *timeout*;	/* **取り込みタイムアウト、秒** */
DWORD * *pStatus*;	/* **ステータス値を受け取る変数のアドレス** */

X11_Acquire 関数は、イメージバッファのフレーム数を取得 (grab) **します。**

パラメータ	記述
hfg	**フレームグラバハンドル。フレームグラバを選択します。**
hbuf	**イメージバッファハンドル。イメージバッファを選択します。X11_Acquire 呼び出し後、イメージバッファのすべてのフレームがデータで埋められます。BUFFER および X11_AllocateBuffer を参照してください。**
timeout	**取り込みタイムアウトの秒数。***timeout* **秒数後に取り込みが完了しない場合は、戻り値を返します。**
pStatus	**ステータス値を受け取る変数のアドレス。イメージバッファのいずれかのフレームの取り込み中に対応する状態が発生すると、個別ビット値が設定されます。**

戻り値

取り込みの完了後、または timeout **の経過後に戻り値を返します。成功すると** 0**、それ以外はエラーコードです。最終フレームの取り込み終了に対応するステータスワードの値で、***pStatus* **のポイントする変数が設定されます。各フレームの取り込みの間で、ステータスビットは、自動的にリセットされません。個別ステータスビットを選択するのに使用する定数の説明については X11_GetStatus を参照してください。**

例

```
PCI pci;
ECODE ecode;
MODE mode = {DEF_MODE};
BUFFER buffer;
DWORD frames = THAT_MANY;
HFG hfg;
float timeout = .5;
DWORD status;

if (!(ecode = X11_InitSystem (&pci))) {
    /* assume we need the 1st frame grabber */
    if (!(ecode = X11_GetHFG (&hfg, pci.PCIslot[0]))) {
        if (!(ecode = X11_AllocateBuffer
                      (&mode, &buffer, frames))) {
```

```
        if (!(ecode = X11_Acquire
             (hfg, buffer.hbuf, timeout, &status))) {
            /* application code here */
        }
        else {
            return ecode;
        }
    }
    else {
        return ecode;
        }
    }
    else {
        return ecode;
    }
}
else {
    return ecode;
}
```

X11_StartAcquire

__declspec(dllimport) ECODE__stdcall X11_SrartAcquire (*hfg, hbuf, acqmode***)**

HFG *hfg*; /* フレームグラバハンドル */
HBUF *hbuf*; /* イメージバッファハンドル */
DWORD *acqmode*; /* 取り込みモード */

X11_StartAcquire 関数は、イメージバッファのフレーム数の取り込みを開始し、即時に戻り値を返します。

パラメータ	記述
hfg	フレームグラバハンドル。フレームグラバを選択します。
hbuf	イメージバッファハンドル。イメージバッファを選択します。X11_StartAcquire 呼び出し後、イメージバッファのすべてのフレームがデータで埋められます。BUFFER および X11_AllocateBuffer を参照してください。
acqmode	取り込みモードスイッチ。次のいずれかになります。

値	記述
AMODE_SINGLE	イメージバッファは一度だけ埋められます。
AMODE_CONT	取り込みが停止するイメージバッファは連続的に埋められます。

戻り値

成功すると 0、それ以外はエラーコードを戻します。取り込みの開始後、即時に返します。アプリケーションは、ステータスビットをポーリングして、または割り込みを使って、取り込みが完了したかどうかを判定します。連続モードが選択されている場合、イメージバッファの最後のフレームが埋められたあと、続いてすぐに、最初のフレームが上書きされます。アプリケーションは、STATUS_READY ビットをポーリング (およびリセット) して、または割り込みを使って、各個別フレームの完了を判定します。AMODE_SINGLE が選択されている場合、イメージバッファの最終フレームが完了した時点で、STATUS_READY_ALL ビットが設定されます。AMODE_CONT が選択されている場合、最初のフレームの取り込み開始時にリセットされるので、STATUS_READY_ALL ビットのポーリングは信頼できません。STATUS_READY_ALL ビットに関連付けされた割り込みはありません。

例

```
PCI pci;
ECODE ecode;
MODE mode = {DEF_MODE};
BUFFER buffer;
DWORD frames = THAT_MANY;
HFG hfg;
DWORD status;
```

```
if (!(ecode = X11_InitSystem (&pci))) {
    /* assume we need the 1st frame grabber */
    if (!(ecode = X11_GetHFG (&hfg, pci.PCIslot[0]))) {
        if (!(ecode = X11_AllocateBuffer
                        (&mode, &buffer, frames))) {
            if (!(ecode = X11_StartAcquire
                  (hfg, buffer.hbuf, timeout, &status))) {
                /* wait until acquisition complete */
                while (!(ecode = X11_GetStatus (hfg, &status)) &&
                        !(status & STATUS_READY_ALL)) {
                }
                if (!ecode) {
                     /* application code here */
                }
                else {
                     return ecode;
                }
            }
            else {
                return ecode;
            }
        }
        else {
            return ecode;
        }
    }
    else {
        return ecode;
    }
}
else {
    return ecode;
}
```

X11_StopAcquire

__declspec(dllexport) ECODE__stdcall X11_StopAcquire (*hfg*)

HFG *hfg*; /* **フレームグラバハンドル** */

パラメータ	記述
hfg	フレームグラバハンドル。フレームグラバを選択します。

X11_StopAcquire 関数は、イメージ取り込みを停止します。X11_StartAcquire を呼び出して取り込みを開始したときに限り、この関数を使用する必要があります。

戻り値
成功すると 0、それ以外はエラーコードを戻します。

X11_GetStatus

__declspec(dllexport) ECODE__stdcall X11_GetStatus (*hfg, pStatus***)**

HFG *hfg*; /* **フレームグラバハンドル** */
DWORD * *pStatus*; /* **ステータス値を受け取る変数のアドレス** */

X11_GetStatus 関数は、フレームグラバステータスワードの値を求めます。

パラメータ	記述
hfg	フレームグラバハンドル。フレームグラバを選択します。
pStatus	ステータス値を受け取る変数のアドレス。

戻り値

成功すると 0、それ以外はエラーコードを戻します。ステータスワードの個別ビットには、次の意味があります。

値	記述
STATUS_READY	フレーム取り込みが完了。アプリケーションによりリセットする必要があります。X11_ResetStatus を参照してください。
STATUS_READY_ALL	イメージバッファの取り込みが完了。最初のフレームの取り込み開始時に自動リセットされます。
STATUS_VIDEO	入力でビデオステータス変更 (例：present から absent へ)
STATUS_HLOCK	入力で水平ロック状態変更
STATUS_OFLOW	オーバフロー検出
STATUS_HSYNC	新しいラインの開始
STATUS_VSYNC	新しいフィールドの開始
STATUS_FMT	ビデオフォーマット変更の検出 (例：NTSC から PAL へ)
STATUS_ERROR	転送エラー発生。これはビットの組み合わせです。

X11_ResetStatus

__declspec(dllexport) ECODE__stdcall X11_ResetStatus (*hfg, mask***)**

HFG *hfg*; /* **フレームグラバハンドル** */
DWORD *mask*; /* **リセットマスク** */

X11_ResetStatus 関数は、フレームグラバステータスレジスタの個別ビットをリセットします。

パラメータ	**記述**
hfg	**フレームグラバハンドル。フレームグラバを選択します。**
mask	**値 1 を指定すると、ステータスレジスタの対応するビットがリセットされます。**

戻り値
成功すると 0、それ以外はエラーコードを戻します。ステータスワードの個別ビットの意味については、X11_GetStatus を参照してください。

X11_GetHFG

__declspec(dllexport) ECODE__stdcall X11_GetHFG (*phfg, slot***)**

HFG * *phfg*;	/* **フレームグラバハンドルのアドレス** */
DWORD *slot*;	/* PCI **スロット番号** */

X11_GetHFG 関数は、指定された、ボードの PCI **スロット番号について、フレームグラバハンドルの値を取り出します。**

パラメータ	記述
phfg	**HFG 型変数へのポインタ**
slot	PCI **スロット番号**

戻り値

成功すると 0**、それ以外はエラーコードを戻します。***phfg* **のポイントする変数をフレームグラバハンドルの値に設定します。**

例

```
PCI pci;
ECODE ecode;
HFG hfg[SYS_GRABBERS];
int i;

if (!(ecode = X11_InitSystem (&pci))) {
    for (i = 0; i < pci.boards; i++) {
        if ((ecode = X11_GetHFG (&hfg[i], pci.PCIslot[i]))) {
            return ecode;
        }
    }
    /* application code here */
}
else {
    return ecode;
}
```

X11_WritePort

__declspec(dllexport) ECODE__stdcall X11_WritePort (*hfg, data, mask***)**

HFG *hfg*; /* フレームグラバハンドルのアドレス */
DWORD *data*; /* 出力ポートに書き込むデータ */
DWORD *mask*; /* 書き込みマスク */

X11_WritePort 関数は、フレームグラバの 4 ビット出力ポートにデータを書き込みます。

パラメータ	記述
hfg	フレームグラバハンドル。フレームグラバを選択します。
data	出力ポートに書き込むデータ。下位 4 ビットのみ有意です。
mask	値 1 を指定すると、対応するビットの修正が可能になります。 他のビットに影響を与えずに個々のビットの修正が可能になります。

戻り値
成功すると 0、それ以外はエラーコードを戻します。

X11_ReadPort

__declspec(dllexport) ECODE__stdcall X11_ReadPort (*hfg, pData***)**

HFG *hfg*; /* フレームグラバハンドルのアドレス */
DWORD * *pData*; /* データ変数のアドレス */

X11_ReadPort 関数は、フレームグラバの 4 ビット入力ポートからデータを読み込みます。

パラメータ	記述
hfg	フレームグラバハンドル。フレームグラバを選択します。
pData	下位 4 ビットにデータを受け取る DWORD 変数へのポインタ

戻り値
成功すると 0、それ以外はエラーコードを戻します。

X11_GetImageSize

__declspec(dllexport) ECODE__stdcall X11_GetImageSize (*pMode, pXsize, pYsize***)**

MODE * *pMode*; /* **MODE 型変数のアドレス** */
DWORD * *pXsize*; /* **水平サイズを受け取る変数のアドレス** */
DWORD * *pYsize*; /* **垂直サイズを受け取る変数のアドレス** */

X11_GetImageSize 関数は、特定モードに対応するイメージの寸法を取り出します。複雑なフォーマッティングオプションを使っていて、イメージ寸法が取り出せないケースで役に立ちます。たとえば、取り出された値を使って、ディスプレイ・ウィンドウを定義することもできます。

パラメータ	記述
pMODE	モードの設定を格納する MODE 型変数へのポインタ
pXsize	ピクセル値でイメージの水平サイズを受け取る DWORD へのポインタ
pYsize	ピクセル値でイメージの垂直サイズを受け取る DWORD へのポインタ

戻り値
成功すると 0、それ以外はエラーコードを戻します。

X11_SetMode

__declspec(dllexport) ECODE__stdcall X11_SetMode (*hfg, pMode***)**

HFG *hfg*; /* **フレームグラバハンドルのアドレス** */
MODE * *pMode*; /* **MODE 型変数のアドレス** */

X11_SetMODE 変数は、要求されたフレームグラバモードを設定します。

パラメータ	記述
hfg	フレームグラバハンドル。フレームグラバを選択します。
pMODE	モードの設定を格納する MODE 型変数へのポインタ

戻り値
成功すると 0、それ以外はエラーコードを戻します。

注記
X11_SetMODE 変数関数呼び出しの結果として、*pMode* **のポイントする MODE 変数の値が変更される場合があります。たとえば、(SCALE8 など)既定のスケール設定の 1 つを使用すると、MODE の advanced 部分のメンバはそれに従って設定されます。**

X11_InterruptOn

__declspec(dllexport) ECODE__stdcall X11_InterruptOn (*pIntData*)

INT_DATA * *pIntData*;　　/* **INT_DATA 型変数のアドレス** */

X11_InterruptOn 関数は、特定フレームグラバについて割り込みを有効にします。

パラメータ	記述
pIntData	割り込みの設定を格納するグローバル変数へのポインタ

戻り値

成功すると 0**、それ以外はエラーコードを戻します。**

注記

SX11.dll は、割り込みの処理として、許可された割り込みが発生したときにウェークアップする並列スレッドを生成します。システム側の割り込み処理プロシージャは、*pIntData* **型のポイントする INT_DATA 型グローバル変数に必要なデータをコピーし、INT_DATA の func メンバのポイントするユーザ関数を呼び出します。INT_DATA 構造体は、次のメンバを格納します。**

メンバ	記述
hfg	フレームグラバハンドル。フレームグラバを選択します。
mask	割り込みマスク。**mask** を 0 に設定すると割り込みが無効になります。次の値に mask を設定すると(論理和を使用できます)、指定の割り込みが有効になります。

値	記述
STATUS_READY	フレーム取り込みが完了
STATUS_VIDEO	入力でビデオステータス変更 (例：present から absent へ)
STATUS_HLOCK	入力で水平ロック状態変更
STATUS_OFLOW	オーバフロー検出
STATUS_HSYNC	新しいラインの開始
STATUS_VSYNC	新しいフィールドの開始
STATUS_FMT	ビデオフォーマット変更の検出 (例：NTSC から PAL へ)
STATUS_ERROR	転送エラー発生。これはビットの組み合わせです。

メンバ	記述
status	フレームグラバのステータス。個別ビットの意味は、割り込みマスクのそれに対応します。割り込み発生時に **status** の値が設定されます。
func	ユーザ割り込み処理関数へのポインタ。関数の戻り値は **DWORD** で、引数はとりません。

priority	**スレッドの優先順位レベルを処理する割り込みを指定する整数。次の値をとります。** **THREAD_PRIORITY_LOWEST** **THREAD_PRIORITY_BELOW_NORMAL** **THREAD_PRIORITY_NORMAL** **THREAD_PRIORITY_ABOVE_NORMAL** **THREAD_PRIORITY_HIGHEST** **THREAD_PRIORITY_TIME_CRITICAL**
total	**X11_InterruptOn 呼び出し後に発生した割り込みの合計数**
lost	**処理されなかった割り込みの数。ユーザ関数の実行中に発生した割り込み**

下記の例に示すように、グローバル変数を経由して、メインアプリケーションとユーザ関数間でデータを相互に受け渡しすることができます。デフォルトで、割り込みはすべて、リセット時にマスクがかけられます。つまり、無効化されます。割り込みのマスクを外して有効にするには、X11_InterruptUnmask **関数を呼び出さなければなりません。また、システム側の割り込み処理プロシージャが割り込みにマスクをかけるので、この関数は、割り込み発生後に呼び出す必要があります。**

例

```
/*  global section */
/*  define USER type to pass data to/from the user interrupt
    handling function.  Assume components' types are defined */
struct USER {
     SOMETYPE user1;
     ANOTHERTYPE user2;
     YETANOTHERTYPE user3;
};
struct USER user;
INT_DATA intdata;
/* end of global section */

ECODE ecode;
HFG hfg;
BUFFER buffer;
BOOL enough;

/* Initialize the system, allocate buffer(s), get handles here */

/* Set up INT_DATA */
intdata.hfg = hfg;
intdata.mask = STATUS_READY;
intdata.func = UserFunc;         //see below;
intdata.priority = THREAD_PRIORITY_NORMAL;
```

```
/* Set up the necessary user data */
user.user2 = NOT_READY;         //some user flag;

/* Enable interrupt */
if (!(ecode = X11_InterruptOn (&intdata))) {
    if (!(ecode = X11_InterruptUnmask (hfg, STATUS_READY))) {
        /* start acquisition */
        X11_StartAcquire (hfg, buffer.hbuf, AMODE_SINGLE);
        /* do whatever is necessary here,
           for example:                  */
        while (!enough) {
           if (user.user2 == READY) {
               user.user2 = NOT_READY;
               /* start next acquisition */
                X11_InterruptUnmask (hfg, STATUS_READY);
                X11_StartAcquire (hfg, buffer.hbuf, AMODE_SINGLE);
               /* image is ready, do something with it */
           }
        }
    }
    else {
        return ecode;
    }
}
else {
    return ecode;
}

DWORD UserFunc (void)
{
    /*  do what you need to here,
        pass the data via USER */
    user.user1 = 1;
    user.user2 = READY;          // etc.
    return 0;
}
```

X11_InterruptOff

__declspec(dllexport) ECODE__stdcall X11_InterruptOff (*pIntData***)**

INT_DATA * *pIntData*;　　　/* **INT_DATA 型変数のアドレス** */

X11_InterruptOff 関数は、特定フレームグラバについて割り込みを無効にします。

パラメータ	記述
pIntData	割り込みの設定を格納するグローバル変数へのポインタ

戻り値
成功すると 0、それ以外はエラーコードを戻します。

注記
X11_InterruptOff 関数の引数は、X11_InterruptOn の呼び出しに使用した引数と同じでなければなりません。

X11_InterruptMask

__declspec(dllexport) ECODE__stdcall X11_InterruptMask (*pIntData***)**

INT_DATA * *pIntData*;　　　/* **INT_DATA 型変数のアドレス** */

X11_InterruptMask 関数は、特定フレームグラバについて割り込みにマスクをかけます。

パラメータ	記述
pIntData	割り込みの設定を格納するグローバル変数へのポインタ

戻り値
成功すると 0、それ以外はエラーコードを戻します。

注記
X11_InterruptMask 関数は、ハードウェアレベルで選択された割り込みを無効化します。割り込み処理プロシージャ自体は引き続きアクティブのままです。X11_InterruptMask 関数の引数は、X11_InterruptOn の呼び出しに使用した引数と同じでなければなりません。

X11_InterruptUnmask

__declspec(dllexport) ECODE__stdcall X11_InterruptUnmask (*pIntData***)**

INT_DATA * *pIntData*; /* **INT_DATA 型変数のアドレス** */

X11_InterruptUnmask 関数は、特定フレームグラバについて割り込みのマスクを外します。

パラメータ	記述
pIntData	割り込みの設定を格納するグローバル変数へのポインタ

戻り値

成功すると 0、それ以外はエラーコードを戻します。

注記

X11_InterruptUnmask 関数は、ハードウェアレベルで選択された割り込みを有効化します。X11_InterruptUnmask 関数の引数は、X11_InterruptOn の呼び出しに使用した引数と同じでなければなりません。

Compact PCI シリーズ

aPCI-8701

取扱説明書

初版発行　2000 年 7 月 26 日

第２版発行　2000 年 8 月 23 日

発行所　株式会社 アドテック システム サイエンス
〒240-0005　神奈川県横浜市保土ヶ谷区神戸町 134
YBP ハイテクセンター 1F
Tel　045-331-7575 ㈹　　Fax　045-331-7770

CPCI-046-000823